Atty. Ref.: 005127.P001

Express Mail No.: EL466330618US

UNITED STATES PATENT APPLICATION

FOR

INTEGRATED INPUT/OUTPUT CONTROLLER

Inventors:

VIRGIL WILKINS
ROBERT HORN
MARC ACOSTA
SANJAY MATHUR

Prepared by:

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard, Seventh Floor
Los Angeles, CA 90025-1026
(714) 557-3800

INTEGRATED INPUT/OUTPUT CONTROLLER

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to concurrent processing and mapping of multiple input/output requests and more particularly to input/output controllers to interface with an array of electronic devices such as computers, servers, networks, storage devices and the like.

2. Background Information

Low-level control of a single disk drive to store data on one or more spinning magnetic disks is well known. Usually a dedicated disk drive controller performs the low level control of moving the read/write heads to the appropriate track and sector to read, write, or format the spinning magnetic disks.

High-level control of a single disk drive is typically provided by software such as a software driver that interfaces the disk drive to a particular operating system of a computer. Another type of high-level control of a single disk drive is partitioning software that partitions a single disk drive into multiple volumes of storage areas.

Another type of high-level control for multiple disk drives is redundant array of individual disks (RAID) software that is

used to provide redundancy and increase reliability of the storage of data when multiple disks are available. Different levels of RAID software are available including RAID 0, RAID 1 and RAID 5, which provide different mechanisms of redundancy and reliability depending upon the system level requirements.

As the amount of data to be stored has become greater and the number of users requiring access to data has become larger, it has become a greater challenge to manage data storage. To ease data storage management, such as data backup and disaster recovery, it has become desirable to centralize data storage to a common pool of disks. The centralized data storage can be made available to multiple users over a network. Furthermore with the advent of the internet, centralized data storage provides increased data availability to a broader and larger distribution of users.

Because the amount of data has grown and the number of users requiring access to centralized data storage has increased, the high level control of multiple disk drives by software alone has become disadvantageous. A main disadvantage to high-level control by software is that it is slow and creates a performance bottleneck in the flow of data into and out of the multiple array of disk drives.

It is desirable to overcome the disadvantages of the prior art.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

Figure 1 is a system block diagram in which the present invention is utilized.

Figure 2 is a functional block diagram of an embodiment of the present invention.

Figure 3 is a process diagram of a Host Interface subsystem of the present invention illustrating input and output packets and tables utilized by a Host Exchange Controller of the Host Interface subsystem.

Figure 4 is a process diagram for the Host Interface subsystem of the present invention illustrating input and output packets utilized by a Command Decode controller of the Host Interface subsystem.

Figure 5 is a process diagram of a Cache Manager of the present invention illustrating the input and output packets and tables utilized by the Cache Manager.

Figure 6 is a process diagram of a Disk Mapping Controller of the present invention illustrating the input and output packets utilized by the Disk Mapping Controller.

Figure 7 is a process diagram of a Disk Exchange Controller of the present invention illustrating the input and output packets utilized by the Disk Exchange Controller.

Figure 8 is a block diagram illustrating fields of a Range Operation Request packet utilized in the present invention.

Figure 9 is a command flow diagram showing a typical sequence of operation of a block read command.

Figure 10 is a command flow diagram showing a typical sequence of operation of a block write command.

Like reference numbers and designations in the drawings indicate like elements providing similar functionality.

BRIEF SUMMARY OF THE INVENTION

The present invention is briefly summarized by the claims that follow below.

DETAILED DESCRIPTION

In the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be obvious to one skilled in the art that the present invention may be practiced without these specific details. In other instances well known methods, procedures, components, and circuits have not been described in detail so as not to unnecessarily obscure aspects of the present invention.

The integrated I/O controller of the present invention integrates host and peripheral connections (such as disk or tape), media protocol controllers (such as Fiber channel, Ethernet or Inifinband protocol management), caching and buffer management hardware, block request mapping algorithms, packet management utilities and an embedded microcontroller. The integrated I/O controller performs resource workload balancing by monitoring a set of operational parameters and routing block level requests to the most appropriate resource. For example, two disk drives may both be able to satisfy the current data request. The integrated I/O controller will dispatch the command to the drive most likely to respond to the request in the least amount of time based upon the monitored parameters. The integrated I/O controller also provides improved fault

tolerance through its integration and reliability as well as its layers of management. The integrated I/O controller provides high-level control of a plurality of disks using hardware circuits in order to increase control efficiency and processing speeds. This high level control of the plurality of disks is integrated into one semiconductor integrated circuit. High-level I/O commands from the server are parsed by the integrated I/O controller and then mapped and routed to the appropriate peripheral (disk, tape, optical drive) in the array of peripherals available to the server. The subsequent data is then routed to/from the Server from/to the designated peripheral by the integrated I/O controller.

In a RAID implementation the integrated I/O controller makes the array of physical disk drives transparent to the server. The array of physical disk drives are grouped together to form larger Logical disk drives and are referenced by the Server as Volumes via a Logical Unit Number (LUN). Algorithms for RAID mapping are executed by the integrated I/O controller for managing data accesses to the Volumes. The integrated I/O controller uses microcoded structures and state machines operating concurrently in much of its circuitry, as opposed to firmware and software that operates sequentially. The microcoded structures and state machines used in the present

invention achieve increased performance while maintaining operational flexibility. A large local buffer is implemented to support the buffering of requested data blocks as well as cached data blocks yet to be requested by the Server or data blocks yet to be written to the physical disks.

Referring now to Figure 1, a system block diagram of a simple storage area network 100 is illustrated. The storage area network 100 is exemplary of a system in which the integrated I/O controller of the present invention is utilized and can be a local area network (LAN), a metropolitan area network (MAN) or a wide area network (WAN). In the storage area network 100, a client array 101 of clients 101A-101N read and write data to at least one disk array 102 having a plurality of disks 103. Between the client array 101 and the disk array 102 resides an I/O controller 110 with RAID functionality to provide centralized data storage and centralized data storage management for the network 100.

In order to access the plurality of disks 103, the clients 101A-101N of the client array 101 are coupled to a network. The clients 101A-101N couple to a hub/switch 106 via cabling or connection 105. One or more servers 108 of a server array 112 couple to the hub/switch 106 as part of the network via cable or

other type of network connection 107. In one embodiment, the hub/switch 106 is an Ethernet switch and the network 100 an Ethernet network. The cabling or connection 105 is usually Ethernet. The one or more servers 108 can include the I/O controller 110. Alternatively the one or more servers 108 can couple through cabling or connection 109 to a Fibre Channel switch 111 and then the I/O controller 110 through cabling or host connection 119 utilizing either Fibre Channel, Ethernet or Infiniband as the protocol for communication. When the switch 111 is utilized to connect the one or more servers 108 to the I/O controller 110, the term Fabric is used to describe this storage area network topology. The I/O controller 110 couples to the plurality of disks 103 and associated disk array 102 by way of cabling or connection 117. The cabling or connection 117 is usually Fibre Channel Arbitrated Loop, but can also be parallel SCSI or Infiniband. The cabling or connection 117 is also referred to herein as the physical media connection interface 117. The plurality of disk 103 maybe optical disks, magnetic disks, semi-magnetic disks, tape drives or other data storage devices. The clients 101A-101N of the client array 101 in one embodiment are personal computers but can be other electronic or electrical devices coupled into a network

configuration for storing and accessing data from the plurality of disks 103.

Referring now to Figure 2, a functional diagram of a printed circuit board (PCB) 200 containing an integrated I/O controller device 210 of the present invention is illustrated. The I/O controller system 110 includes the PCB 200. As illustrated in Figure 2, the PCB 200 includes the integrated I/O controller device 210, a host connector 201, at least one or more disk connectors 202, a cache buffer memory 204 a cache state table buffer memory 203, a system micro-processor 205, and program memory (such as FLASH, ROM, and SRAM) 206 for storing the microprocessor program, configuration tables and other program structures. The integrated I/O controller device 210, a single silicon integrated circuit, couples to the components of the printed circuit board 200 as shown in Figure 2. The host connector 201 is for coupling the I/O controller device 210 to one or more servers 108 of the server array 112 via the network switch 111. The plurality of disk connectors 202 are for coupling the integrated I/O controller device 210 to at least one disk array 102 having the plurality of disks 103. The cache buffer 204 provides temporary storage for all data blocks transferred to/from the plurality of Servers 112 and the plurality of disks 103. The cache state table buffer memory 203

is utilized to maintain status of each block of data retained in the cache buffer memory 204. The microprocessor 205 and program memory 206 are utilized to assist the integrated I/O controller device 210 in its initialization and configuration of the storage or disk array 102 for access of data to the plurality of disks 103.

Figure 2 also illustrates a functional block diagram of the integrated I/O controller device 210 which includes a host interface sub-system 220, a disk interface subsystem 230, a buffer manager 270, a cache manager 260, a disk mapping controller 250 and a micro-controller subsystem 240. The micro-controller subsystem 240 is coupled together with the host interface subsystem 220 and the disk interface subsystem 230 via the packet/control bus 214 to perform initialization and error handling at these interfaces. The micro-controller subsystem 240 is also coupled together with the buffer manager 270, the cache manager 260, and the disk mapping controller 250 in order to control the flow of non-user data information from the plurality of servers 108 of the server array 112 and the plurality of disks 103 of the disk array 102.

The buffer manager 270 couples to the cache buffer memory 204. The buffer manager 270 additionally couples to the disk

interface subsystem 230 and the host interface subsystem 220 by way of data channels 212 for the transfer of data to/from the buffer memory 204 from/to a host server 108 or a disk 103 of the disk array. The buffer manager 270 uses traditional arbitration methods to arbitrate between multiple concurrent access requests from/to the host interface subsystem 220, disk interface subsystem 230 and micro controller subsystem 240 to/from the cache buffer memory 204.

The cache manager 260 couples to the cache state table buffer 203 of the cache memory and manages the coherency of data blocks in the cache buffer memory 204. The cache memory for the cache buffer memory 204 and the cache state table buffer 203 may be a dynamic random access memory or other type of memory. Additionally, the cache buffer memory 204 and cache state table buffer 203 could be integrated into the integrated I/O controller device 210 but it is preferable that it be external in order to provide system flexibility.

The host interface subsystem 220 of the integrated I/O controller device 210 includes a Host Exchange Controller (HEC) 222 and a Command Decode Controller (CDC) 224. The host exchange controller 222 controls the physical connection and protocol between the one or more servers 108 of the server array

112 and the integrated I/O controller device 210. This host connection or attachment 119 to the servers 108 can be Fibre Channel, Ethernet, PCI, PCI-X or Infiniband in alternate embodiments. The command decode controller 224 parses all commands received from the one or more servers 108 of the server array 112 and extracts the data block level read, write requests while routing all other commands to the micro controller subsystem 240. All commands over the host connection are preferably in the form of the Small Computer System Interface (SCSI) Command Descriptor Block (CDB) standard.

The disk interface subsystem 230 controls the connections between the plurality of disks 103 of the disk array 102 and the integrated I/O controller device 210 and includes one or more Disk Exchange Controllers 232A-232N. A preferred embodiment of the disk interface subsystem executes the industry standard fibre channel connection protocol (FC-AL, FC-AL2 or FC-AL3) on a physical level. To send and receive commands and data with the plurality of disks 103, the integrated I/O controller device 210 uses the SCSI over Fibre Channel protocol (such as FCP, FCP2 or SCSI-3).

The micro-controller subsystem 240 handles non-data flow commands, error and exception handling events as well as system initialization.

Referring now to Figure 3, a process diagram of the Host Exchange Controller (HEC) 222 is illustrated. The host exchange controller 222 receives command and data frames from the physical media interface (Fibre Channel, Ethernet, Infiniband). Data frames received by the host exchange controller 222 are routed to the buffer manager 270 over the data channels 212. The command frames received by the host exchange controller 222, such as command frames 301 illustrated in Figure 3, are stored as command entries into a host exchange table 303. The host exchange table 303 in one embodiment can hold thousands of command entries.

Each command entry 313 in the host exchange table 303 has a tag number 311 which is either assigned or implied. In one embodiment, the tag number 311 is the same as the entry number, (i.e. the physical position of the entry), in the host exchange table 303. The physical location of the command entry in the host exchange table 303 implies the tag number and thus an actual tag number field 311 need not be stored with each entry.

Each entry 313 in the host exchange table also includes several other fields to properly track the progress of the

command through the integrated I/O controller device 210 and subsequently transfer data blocks to/from the requesting server 108. Field 305 contains network context information about the command which includes the Servers address on the network so response data can be routed properly back to the requesting server. Field 307 contains the SCSI CDB. Field 309 contains buffer pointers to physical locations in the cache buffer 204 where the data will be stored. The Cache Manager 260 assigns the proper values and stores them in field 309 of each entry 313 when the buffer space is physically allocated. Field 311 represents the unique tag number associated with this host exchange entry 313.

After the host exchange controller 222 creates the command entry 313 for a received command packet 301, a tag packet 315 containing the tag number for the received command is formed by the host exchange controller. The tag packet 315 is then sent to the Command Decode Controller 224 of host interface subsystem 220.

Referring now to Figure 4, a process diagram for the Command Decode Controller 224 is illustrated. The command decode controller 224 receives the tag packets 315 including the tag numbers pointing to SCSI CDBs' that need further processing. When its ready to process the next command, the command decode

controller 224 reads the command entry 313 associated with a tag packet 315, extracts the SCSI CDB from it, and validates the contents of the command entry. The command decode controller 224 includes hardware circuitry and a microcoded state-machine that specializes in decoding and queuing dataflow requests, such as block read and write commands. All commands which are received by the command decode controller 224 and determined to be valid, are placed into a pool or table of command queues 403. A separate command queue 403A-403N is maintained in the pool or table of command queues 403 for each volume of the disk array 112 that is accessible to the server array 112. The command decode controller 224 implements the industry defined SCSI Architectural Model (SAM) for command processing, including command ordering and queuing. If implemented as a RAID controller, the command decode controller 224 translates the Logical Block Address (LBA) of the requested volume into a Global (or generic) logical block address (GLBA) understood by other circuitry in the integrated I/O controller device 210. This GLBA along with a sector count from the SCSI CDB, the direction of the requested transfer (read or write), and tag number 311 are combined together to form a Range Operation Request (Ror) 801 which is further described below and illustrated in Figure 8. The Ror 801 is combined with a flag

field 405 to form an output packet 401 from the command decode controller 224 which is sent to the Cache Manager 260. In one embodiment of the invention the flag field 405 includes information about the source of the packet as well as routing information defining the destination or recipient of the packet. The flag field 405 can be expanded to include additional information about the process state of the command entry 313 to assist the micro-controller subsystem 240 in the event of an error condition or in the development and debug of the integrated I/O controller device 210. Any task management function required by an incoming host command, such as a request for the capabilities and configuration of the integrated I/O controller via SCSI Mode pages, is routed by the command decode controller 224 to the micro-controller subsystem 240.

Referring now to Figure 8, the format of the Range Operation request (Ror) packet 801 is illustrated. The Ror packet 801 is used to communicate command requests between functional blocks of the integrated I/O controller device 210. The Ror packet 801 includes a field 803, a field 805, a field 807 and a field 809. Field 803 contains the starting GLBA for the range of blocks to be operated upon. Field 805 of the Ror 801 contains the number of blocks to be operated upon (range) which depends upon if the Ror is generated by the cache manager

260, the command decode controller 224 or another functional blocks of the integrated I/O controller device 210. If the command is generated by the command decode controller 224, the value of the range (i.e. number of blocks to be operated upon) is generated from the sector count of the SCSI CDB. If the command is generated by the Cache Manager 260, the value of the range (i.e. number of blocks to be operated upon) represents the number of blocks in the cache buffer 204 to which access is desired in order to read or write data blocks to/from the disk array 102 and the plurality of drives 103. Field 807 indicates the type of range operation being requested, either a Read data operation or a Write data operation. Field 809 contains the tag number 311 from which the given Ror 801 is derived or associated.

Referring now to Figure 5, a process diagram for the Cache Manager 260 is illustrated. The cache manager 260 manages the state of all blocks within the cache buffer memory 204. The cache manager receives an input packet 401 including an Ror packet 801 and flags 405. The Ror 801 is extracted from the input packet 401 by the cache manager 260 and the range field 805 is read. The cache state buffer 203 includes a cache state table 501 which has entries each being associated with the data blocks stored into the cache buffer 204. The cache manager 260

then searches the entries in the cache state table 501 to determine if there are blocks of data in the cache buffer 204 which overlap the range of blocks specified by the range field 805 in the Ror 801. In the case of a read request, if the range of blocks requested by the command exists in the cache buffer 204 (i.e., a cache hit), the cache manager 260 updates the buffer pointer field 309 of the respective command entry 313 associated with this Ror 801 and then generates a response packet 505 which is sent to the host exchange controller 222. The response packet 505 includes a data valid field 507 indicating that data is valid and ready for transfer from the cache buffer 204 and the tag number 311. If the requested range of blocks are not detected in the cache buffer 204 for either a read or write request (i.e., a cache miss), the cache manager allocates buffer space in the cache buffer 204, updates the buffer pointers 309 in the host exchange entry 313 accordingly, and generates an output packet 503 including a new Ror 801 indicating the range of blocks required to satisfy the original range request of packet 401. The output packet 503 is passed to the disk mapping controller 250 for further processing to gain access to the disk array 112 because the desired data blocks were not in the cache buffer 204. The output packet 503 also contains a flag field 509 similar in function and capability to

the earlier described flag field 405 of the packet 401 from the command decode controller 224. Using caching algorithms (such as Least Recently Used (LRU), Pseudo Least Recently Used (PLRU), or Most Recently Used (MRU)) along with a data aging (such as time stamps on the blocks of data) technique, the cache manager 260 can determine which blocks within the cache buffer 204 should be retained and which blocks should be removed or overwritten first. The cache manager is not limited to these specified caching algorithms nor is it required that the cache manager perform a caching algorithm to determine how to retain or overwrite data in embodiments of the invention.

Referring now to Figure 6, a process diagram for the Disk Mapping Controller (DMC) 250 is illustrated. Generally, the disk mapping controller 250 converts requests for global logical block addresses (GLBAs) and maps them into physical block addresses. It is possible that a global logical block address can be mapped into more than one physical data block address when a redundant copy exists or data mirroring is utilized in a RAID controller application. If more than one source for the physical data block requested exists, the disk mapping controller 250 determines the best source of the data by evaluating a set of parameters and gathered statistics. These parameters and statistics are part of a cost function table 607

and can be used by the disk mapping controller to perform load balancing over the plurality of disk drives 103 of the disk array 102. In other embodiments of the present invention, the load balancing function need not be implemented. In a RAID controller application of the integrated I/O controller device 210, the disk mapping controller 250 implements in hardware and microcoded state-machines algorithms to provide RAID mapping of data blocks across the plurality of drives 103 of the disk array 102. RAID 0, RAID 1, RAID 0+1, RAID 5 and RAID 50 are all implemented and supported by the microcoded state-machine within the disk mapping controller 250. After calculating the required physical drive or drives to satisfy the Ror 801 of packet 503, the disk mapping controller 250 generates an output packet 601. The output packet 601 includes a Basic Drive Command 603 and the tag number 311 of the original host command requesting the data and is also referred to herein as a basic command packet 601. The disk mapping controller 250 generates output packets 601 which are sent to each of the Disk Exchange Controllers 232A-232N corresponding to each physical drive requiring access to satisfy the Ror of the input packet 503.

Alternatively, a linked list or a series of drive commands 615 can be formed, chained together and stored in a memory for later use when needed. The Cache Buffer memory 204 can be used

to store this linked list of drive commands 615 as illustrated in Figure 6. In case of the linked list of drive command 615, the disk mapping controller 250 sends a chained command packet 609 which includes a chained command pointer 611 and the tag number 311 to the disk exchange controllers 232A-232N. The chained command pointer 611 points to the list of drive commands 615 in the memory for processing by the Disk Exchange Controllers 232A-232N.

Figure 7 is a process diagram for each of the Disk Exchange Controllers (DEC) 232A-232N. Each disk exchange controller 232A-232N receives either a basic command packet 601 or alternatively a chained command packet 609 if chained commands are used. The basic command packet 601 includes the tag number 311 and a basic drive command 603. The chained command packet 609 includes the tag number 311 and the chained command pointer 611. If a disk exchange controller 232A-232N receives the basic command packet 601, it converts the basic drive commands 603 into SCSI command descriptor blocks (CDB) and then encapsulates them into an appropriate frame or output packet format 701 for transmission over the physical media connection interface 117 to the disks 103 of the disk array 102. If a disk exchange controller 232A-232N receives the chained command packet 609, it reads the chained command pointer accesses a stored command in a chain and

converts it from a basic drive command into a SCSI command descriptor blocks (CDB) which is then encapsulated into an appropriate frame or output packet format 701.

In one embodiment, each disk exchange controller 232A-232N implements the defined protocol for a SCSI initiator as documented in the SCSI Architectural Model specification (SAM) which is further refined by the SCSI over Fibre Channel (FCP) and Private Loop Direct Attach (PLDA) standards specifications for Fibre Channel implementations in generating the peripheral command packet or frame 701. Similarly in embodiments of the invention supporting Ethernet (SCSI over Internet Protocol) and Infiniband, each disk exchange controller 232A-232N is microcoded to satisfy the requirements of these protocols and generate the appropriate peripheral command packet or frame 701. When data is transmitted from the cache buffer 204 to the plurality of disk drives 103 of the disk array 102 as opposed to commands, the disk exchange controller 232A-232N encapsulates the data with the appropriate frame or packet format for the physical media connection interface 117 utilized. When data is received from the plurality of disk drives 102 the disk exchange controller removes any data encapsulation information required by the physical media interface connection 117 and routes the data to the cache buffer 204. The buffer pointers 309, in the

entry 313 specified by the tag number 311 associated with this operation, are utilized to specify the location in the cache buffer 204 to write the data.

Referring now to Figure 9, a flow diagram showing the sequence of operations for processing a host read command is illustrated. A host command packet 301, including the host read command, is received by the host exchange controller 222 and stored in the host exchange table 303. The host exchange controller 222 then passes a tag number 311 in a tag packet 315 to the command decode controller 224 which indicates where the new SCSI CDB 307 is located in the host exchange table 303. The Command Decode Controller 224 extracts and processes the SCSI CDB 307 as previously explained in the description of Figure 4. The command decode controller 224 then passes a packet 401 to the Cache Manager 260. The cache manager 260 operates on the packet 401 as previously explained in the description of Figure 5. Any requested data not currently located in the cache buffer 204 is requested from the Disk Mapping controller 250 by the generation of a new packet 503 including a new Ror 801. The disk mapping controller 250 operates on the packet 503 as previously described in the description of Figure 6 and passes the requests for data to one of the Disk Exchange Controllers 232A-232N by sending basic command packets 601 or the alternative chained

command packet 609. The disk exchange controller 232A-232N receiving the command packet acts as a SCSI initiator and manages the sending of one or more disk commands to the plurality of disk drives 103 and then subsequently manages the transfer of data from the disks to the cache buffer 204. When all data has been received from the plurality of disk drives 103 of the disk array 102, the Cache Manager 260 updates the state of the blocks in the Cache State Table 501. The cache manager 260 then notifies the host exchange controller 222 that data is valid in the cache by sending the valid packet 505. Upon receipt of the valid packet 505, the host exchange controller 222 initiates the transfer of data from the location specified in the cache buffer 204 by the buffer pointers 309 listed in the entry 313 in the host exchange table 303 for this command, to the server specified in the network context information field 305 in the same entry 313.

Referring now to Figure 10, a flow diagram showing the sequence of operations for processing a host write command is illustrated. A host command packet 301, including the host write command, is received by the Host exchange controller 222 and stored in the host exchange table 303. The host exchange controller 222 then passes a tag number 311 in a tag packet 315 to the command decode controller 224 which indicates where the

new SCSI CDB 307 is located. The Command Decode Controller 224 extracts and processes the SCSI CDB 307 as previously explained in the description of Figure 4. The command decode controller 224 then passes a packet 401 including Ror 801 to the Cache Manager 260. The cache manager 260 examines the Ror 801 of the packet 401 to determine how much buffer or storage space to allocate in the cache buffer 204 for processing the data. Once enough buffer space has been allocated to process the Ror 801, the cache manager 260 updates the buffer pointers 309 in the respective entry 313 in the host exchange table 303 as referenced by the tag number 311. The cache manager 260 then sends a valid packet 505 indicating that sufficient buffer space is available to receive data from the host. If write caching is disabled, these blocks of data will be immediately processed for transfer to the plurality of disk drives 103 of the disk array 102. The write operation proceeds by generating a new packet 503 to the Disk Mapping Controller 250 indicating the range of data to be written from the cache buffer 204. The disk mapping controller 250 operates on the packet 503 as previously explained in the description of Figure 6 and passes the requests to write data to one of the Disk Exchange Controllers 232A-232N by sending basic command packets 601 or the alternative chained command packet 609. The disk exchange controller 232A-232N

receiving the command packets acts as a SCSI initiator and manages the sending of one or more disk commands to the plurality of disk drives 103 and then subsequently the transfer of data from the cache buffer 204 to the disk drives 103 of the disk array 102. When all data has been transferred to the plurality of disk drives 103 of the disk array 102, the Cache Manager 260 updates the state of the blocks in the Cache State Table 501.

The present invention has many advantages over the prior art. One advantage of the present invention is that data communication rates are increased by integrating the I/O controller in hardware. Another advantage of the present invention is that data throughput is also increased by providing RAID mapping control in hardware. Still another advantage of the present invention is that the RAID mapping is flexible through programmability

While certain exemplary embodiments have been described and shown in the accompanying drawings, it is to be understood that such embodiments are merely illustrative of and not restrictive on the broad invention, and that this invention not be limited to the specific constructions and arrangements shown and described, since various other modifications may occur to those ordinarily skilled in the art. For example, the integrated I/O

controller has been described as interfacing to a disk array of a plurality of disks but can also interface to other storage devices or peripheral devices to provide I/O control. Additionally, it is possible to implement the present invention or some of its features in hardware, firmware, middleware, software or a combination thereof where one or more instructions are provided in a processor readable storage medium such as a magnetic, optical, or semiconductor storage medium.